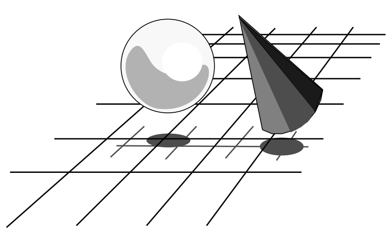
OpenGL and X

X TECHNICAL CONFERENCE 94

A TECHNICAL OVERVIEW OF OpenGL™ and the X Window System®



Publication Number: OGL-1B
Publication Date: 1/94

ABSTRACT

These notes accompany a tutorial of OpenGL and the X Window System to be delivered at the X Technical Conference 94.

NOTICES

Speakers

- Mason Woo
  - Product Manager, OpenGL, Silicon Graphics
  - co-author of *OpenGL Programming Guide*
  - morning session (What OpenGL is)
- Mark J. Kilgard
  - Member of Technical Staff, Silicon Graphics
  - author of OpenGL & X articles in *The X Journal*
  - afternoon session (OpenGL and X)

Goals of OpenGL

- Allow construction of portable and interoperable 3D graphics programs
- Avoid subsetting
- Sophisticated graphics, from workstations to PCs
- Vendor neutrality

What is OpenGL?

- It's a rendering library
- A layer of abstraction between graphics hardware and an application program
- An API to produce high-quality, color images of 3D objects (group of geometric primitives) and 2D primitives (bitmaps and raster rectangles)
- Window System and Operating System independent
    - use with X Window System under UNIX®
    - or Windows NT®
    - or OS/2®
- 2D and 3D graphics functions
- A State Machine

Getting Your Hands on OpenGL

- OpenGL libraries obtained from licensees
  - licensees build the extension
  - Silicon Graphics grants OpenGL licenses
  - application programmers do not need to become licensees
- Licensees don’t need much hardware
  - Sample Implementation designed to run with
    - a generic CPU
    - a simple frame buffer
    - hardware support for rasterization (line and polygon scan conversion)
- But if they have the hardware
  - they can optimize OpenGL for it

Who's Who of OpenGL

- Cray Research (client side only)
- Digital Equipment
- Du Pont Pixel Systems (Sun)
- Evans and Sutherland
- Hitachi
- IBM
- Intel
- Intergraph
- Kubota Pacific
- Media Vision (PC graphics); formerly Pellucid
- Microsoft (Windows NT platforms)
- NEC
- Portable Graphics (Sun as layer on XGL; HP as layer on Starbase)
- Silicon Graphics
- SONY

OpenGL API Hierarchy

GL Utilities (glu) circles, arcs, NURBS, etc.	GLX Utilities (glx) Context, Visuals, Swapbuffers, Synchronization
OpenGL Base (gl)	X Windows (Xlib) Window maint., Colormaps, Event Handling, Pixmaps, Text

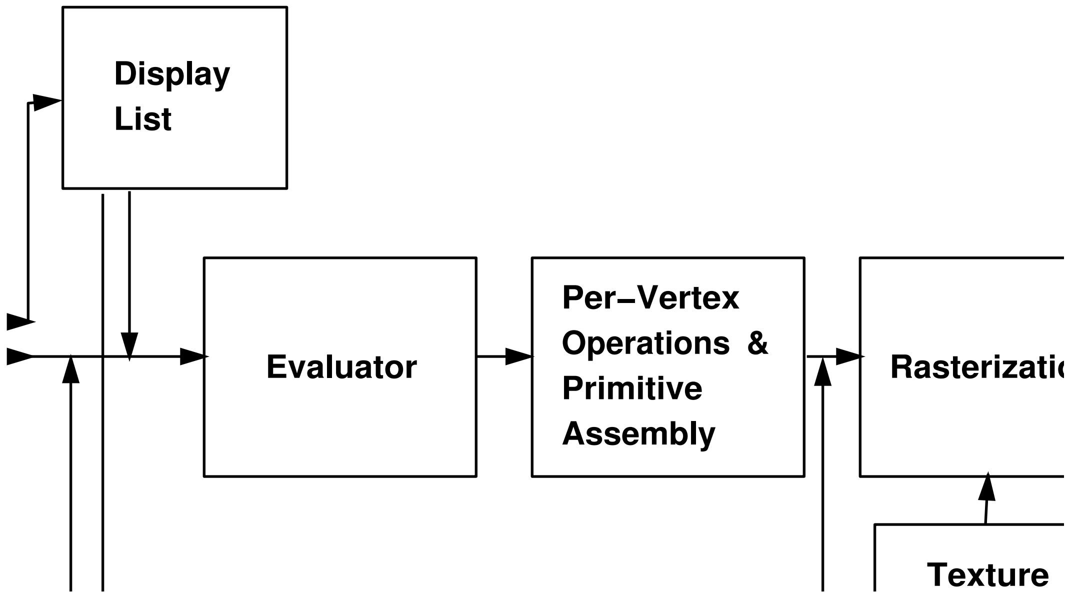
OpenGL State Machine

- Drawing Geometry and Clearing the Screen
- Point, Line, and Polygon Attributes (Size, Width, or Stipple)
- Images and Bitmaps
- Transformations
- Colors and Alpha Blending
- Antialiasing
- Lighting and Texturing
- Fog and Depth Cueing
- Hidden Surface Removal
- Accumulation Buffer
- Stencil Planes
- Feedback and Selection
- Evaluators
- Display Lists and Text

Utility Library (GLU)

- Utility library is a set of commonly used graphics routines
  - built on top of OpenGL
- Quadric surface routines
  - spheres, cones, open cylinders, and tesselated disks for circles and arcs
- Polygonal surface routines
  - concave polygons, polygons with holes, etc.
- NURBS (with trimming)
- Rendering directives
  - outlining, culling, sampling for polygon tessellation
- Picking

OpenGL State Machine Diagram



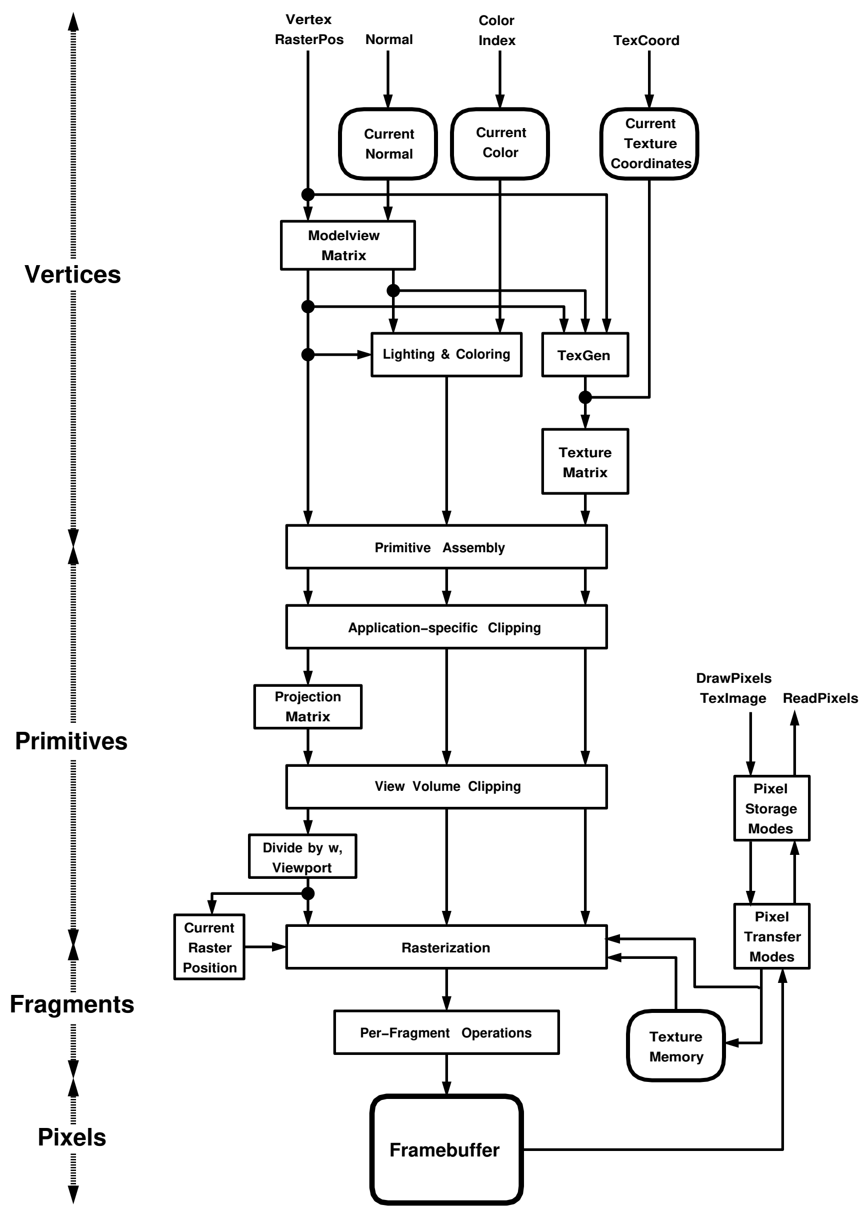
OpenGL Main Points

- The vertex is the fundamental primitive
- As a primitive is drawn, each of its vertices is affected by the current "state" variables:
  - transformation matrices, color, lighting, texture, fog, rasterization, etc.
- All operations are "really" 3-D
- When window contents are damaged, redraw entire contents
- The frame buffer is a useful resource for hidden surface removal (depth buffer), motion blur and depth of field effects (accumulation buffer), and other effects

OpenGL Main Points (continued)

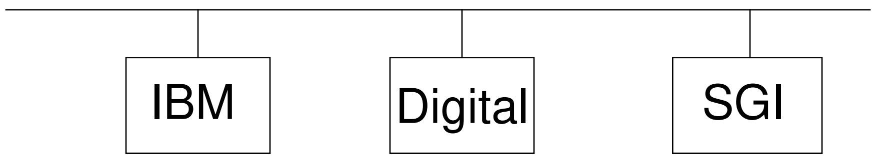
- Display lists are for caching and efficiency, but they are not mandatory. Display lists *may* reside on the server.
- OpenGL does NOT perform non-rendering operations which would be redundant with the window system: window management, event handling, color map operations, etc. Use Xlib for those.

OpenGL Data Flow Diagram



OpenGL extensions to X (GLX)

- GLX extended servers have visuals which support OpenGL rendering
- Interoperability--OpenGL extended protocol



Special OpenGL visuals

- at least two visuals supported
- at least one color buffer (double buffering optional)
- a stencil buffer of at least 1 bit
- a depth buffer of at least 12 bits
- RGBA visual with
  - an accumulation buffer
  - color buffer size must be as great as that of the deepest TrueColor, DirectColor, PseudoColor, or StaticColor visual
- color index visual with
  - as many color bitplanes as the deepest PseudoColor or Static Color visual

Mixed Model

- What is a Mixed Model program?
  - uses the OpenGL and X Window System routines within the same program and inside the same window.
  - uses the OpenGL for rendering

    2-D and 3-D wireframe and filled geometry

    2-D and 3-D transformations

    lighting, shading, and texturing
  - uses the X Window System for window management, input handling, color management, menus

    can use Xlib and/or Xt with a widget set
- This afternoon, Mark Kilgard will discuss this

OpenGL Command Syntax

glVertex3fv

v indicates vector format, if present

data type: f float

d double float

s signed short integer

i signed integer

number of components (2, 3, or 4)

- Other data types in other OpenGL commands
  - b character
  - ub unsigned character
  - us unsigned short integer
  - ui unsinged integer
- scalar and vector formats

States

- `glEnable (GLenum capability)`
- `glDisable (GLenum capability)`
- `GLboolean glIsEnabled (GLenum cap)`
  - turn on and off OpenGL states
  - capability can be one of (partial list):

    `GL_BLEND` (alpha blending)

    `GL_DEPTH_TEST` (depth buffer)

    `GL_FOG`

    `GL_LIGHTING`

    `GL_LINE_SMOOTH` (line antialiasing)

Querying States

- `glGet*()`

```
glGetBooleanv(GLenum pname, GLboolean *params)

glGetIntegerv(GLenum pname, GLint *params)

glGetFloatv(GLenum pname, GLfloat *params)

glGetDoublev(GLenum pname, GLdouble *params)
```

man page is 14 pages long

number of color bits: GL_ALPHA_BITS, GL_BLUE_BITS,

GL_GREEN_BITS, GL_RED_BITS, GL_INDEX_BITS,

GL_DEPTH_BITS, GL_ACCUM_*_BITS

Drawing Geometry

- `glBegin (GLenum primitiveType)`
- `glEnd ()`
  - where primitiveType is `GL_POINTS, GL_LINE_STRIP, GL_LINE_LOOP, GL_LINES, GL_POLYGON, GL_TRIANGLES, GL_QUADS, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN, GL_QUAD_STRIP`
- Command syntax for drawing geometry

```
glBegin (primitiveType);
glVertex* (coordinates);
          .
glVertex* (coordinates);
glEnd ();
```

- can put between `glBegin()` and `glEnd()`

`glVertex`- vertex position

`glColor`- current color

`glIndex`- current color index

`glNormal`- current surface normal (lighting)

`glMaterial`- current material property (lighting)

Images and Bitmaps

- In addition to geometry, OpenGL supports pixel data
- Bitmap (a single bit per pixel)
  - for characters in fonts

`glRasterPos*()` specifies a position for a bitmap

`glBitmap()` renders a bitmap

- Image (typically many bits of data per pixel)

`glReadPixels()`, `glDrawPixels()`, and `glCopyPixels()` manipulate rectangles of pixel data

  - may be zoomed
  - pixel storage and transfer modes (e.g., good for endian reversal)

Point, Line, and Polygon Attributes

- Point size

`glPointSize (GLfloat size)`

- Line width and stipple

`glLineWidth (GLfloat size)`

`glLineStipple (GLint factor, GLushort pattern)`

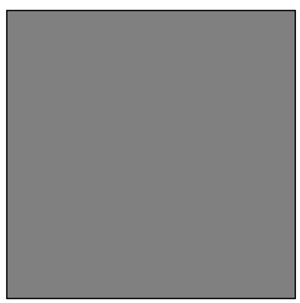
- Polygon stipple and shading

`glPolygonStipple (const GLubyte *mask)`

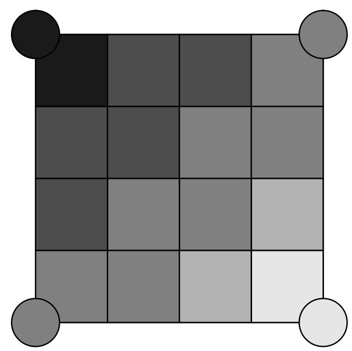
`glShadeModel (mode)` where `mode` is `GL_FLAT` or `GL_SMOOTH`

- Primitives shaded with one color (flat) or a spectrum of adjacent colors (smooth)

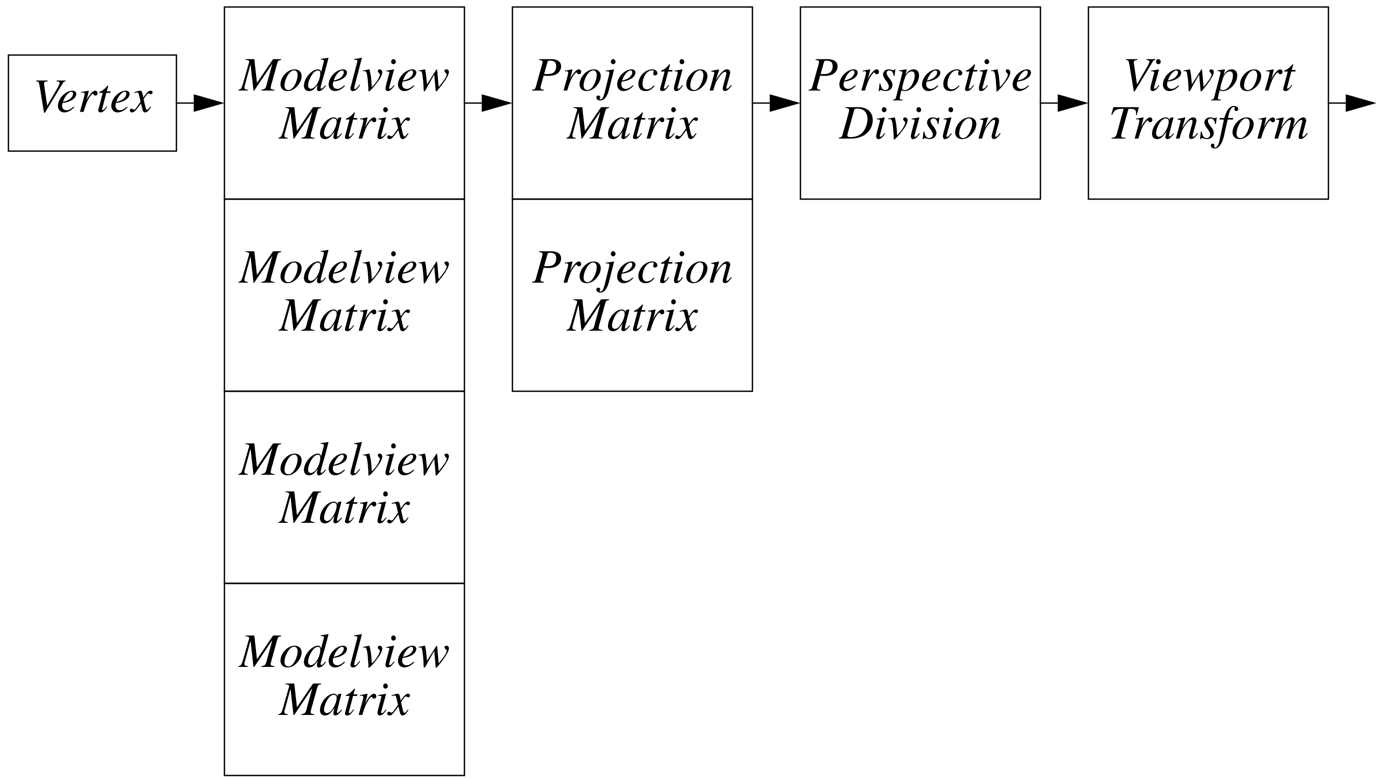
Flat shading



Smooth shading



Transformation Flow

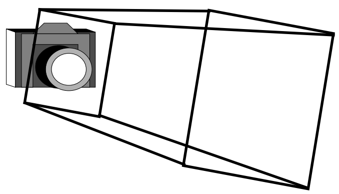



```
glMatrixMode(mode), glLoadIdentity(), glPushMatrix(),
glPopMatrix(), glLoadMatrix(), glMultMatrix()
```

Camera Analogy

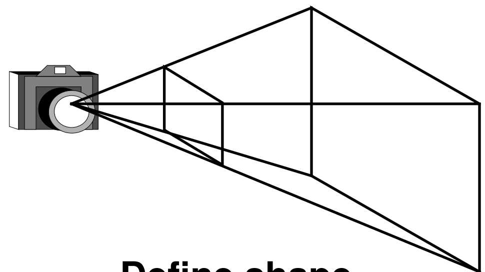
Transformation

viewing



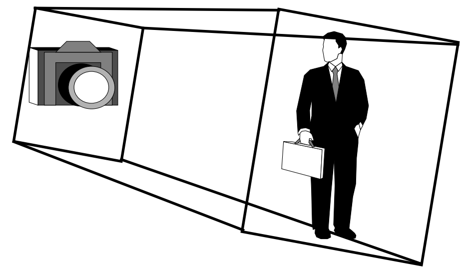
Define position of the viewing volume in the world

projection



Define shape of viewing volume

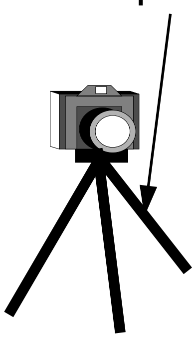
modeling



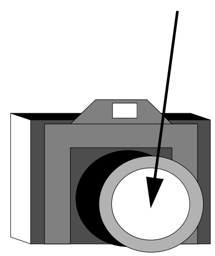
Define position of the models in the world

Camera

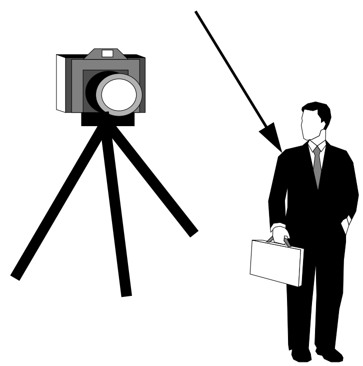
tripod



lens



model



Transformations

- Projection Matrices
  - perspective or orthographic parallel projection

```
glFrustum(left, right, bottom, top, near, far)
glOrtho(left, right, bottom, top, near, far)
gluPerspective(fovy, aspect, zNear, zFar)
gluOrtho2D(left, right, bottom, top)
```

- Viewing (move camera position)

```
gluLookAt(eyex, eyey, eyez, centerx, centery, centerz,
     upx, upy, upz)
```

- Modeling (move local coordinate system)

```
glTranslate{fd}(x, y, z)
```

`glRotate{fd}(angle, x, y, z)`--arbitrary axis of rotation

```
glScale{fd}(x, y, z)
```

- Screen clipping

```
glViewport(x, y, width, height)
```

Before We Look at Some Code

- Some other OpenGL routines

```
glClearColor (r,g,b,a)
```

  - choose RGBA value for clearing color buffer

```
glClear (bitfield)
```

  - set bitplane area of the viewport to currently selected values

- aux routines are not OpenGL routines
  - hides other routines; makes examples shorter
  - initialize and open window, handle keyboard, mouse, and redraw events, enter event-driven loop, and draw 3D models (spheres, cones, cylinders, torii, etc.)

```
auxInitDisplayMode (modes);
auxInitPosition (left, bottom, width, height);
auxInitWindow (titleString); auxReshapeFunc (func);
auxMainLoop (displayFunc); auxSolidSphere (radius);
```

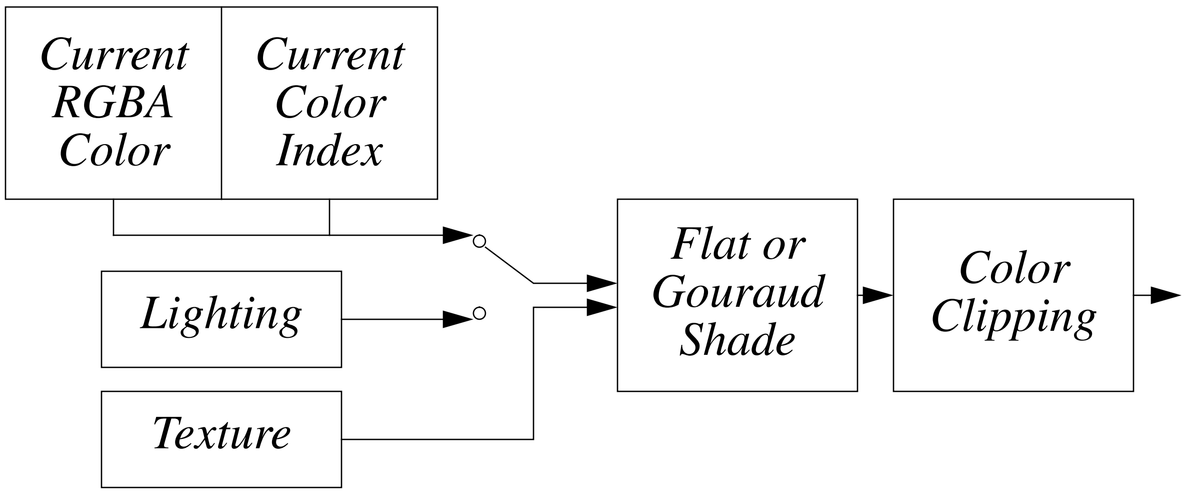
An OpenGL Program

```
/*
 * smooth.c
 * This program demonstrates smooth shading.
 * A smooth shaded polygon is drawn in a 2-D
 * projection.
 */
#include <GL/gl.h>
#include <GL/glu.h>
#include "aux.h"

/*  GL_SMOOTH is actually the default shading
 *  model.
 */
void myinit (void)
{
    glClearColor (0.0,0.0,0.0,0.0);
    glShadeModel (GL_SMOOTH);
}

void triangle(void)
{
    glBegin (GL_TRIANGLES);
```

```
    glColor3f (1.0, 0.0, 0.0);
    glVertex2f (5.0, 5.0);
    glColor3f (0.0, 1.0, 0.0);
    glVertex2f (25.0, 5.0);
    glColor3f (0.0, 0.0, 1.0);
    glVertex2f (5.0, 25.0);
    glEnd ();
}

void display(void)
{
    glClear (GL_COLOR_BUFFER_BIT);
    triangle ();
    glFlush ();
}

void myReshape(GLsizei w, GLsizei h)
{
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if (w <= h)
          gluOrtho2D (0.0, 30.0, 0.0,
          30.0 * (GLfloat) h/(GLfloat) w);
    else
          gluOrtho2D (0.0, 30.0 * (GLfloat) w
```

```
          /(GLfloat) h, 0.0, 30.0);
    glMatrixMode(GL_MODELVIEW);
}

/*  Main Loop
 *  Open window with initial window size, title
 *  bar, RGBA display mode, and handle input
 *  events.
 */
int main(int argc, char** argv)
{
    auxInitDisplayMode (AUX_SINGLE | AUX_RGBA);
    auxInitPosition (0, 0, 500, 500);
    auxInitWindow (argv[0]);
    myinit();
    auxReshapeFunc (myReshape);
    auxMainLoop(display);
}
```

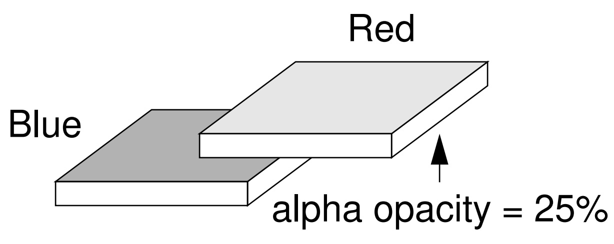
Processing of Colors



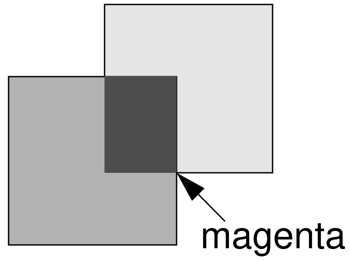
- Color is modal (RGBA or color index)
  - must be set upon window initialization
- Determine current color (RGBA)
  - if lighting is enabled, use lighting to compute color
  - else use currently set RGBA or color index value
  - then apply antialiasing, alpha blending, texturing or other color operation
- Loading color map (look up table) is a window system operation

Alpha Blending

- Translucency effects
- 0 % to 100 % opacity



Side View



Top View

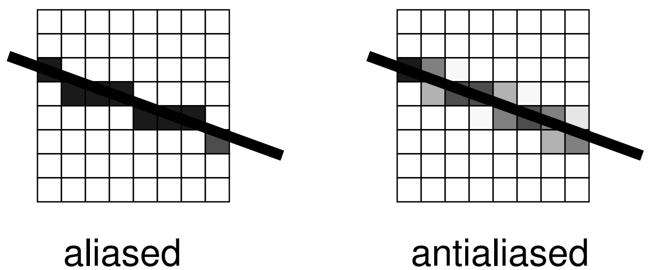
- order of drawing is important
- don't need alpha buffer to do translucency

```
glBlendFunc (GLenum srcFactor, GLenum destFactor)
```

Antialiasing

- Cures the "Jaggies"; creates smooth points & lines

- Lines redrawn 2 or 3 times
- Pixel averaging algorithm

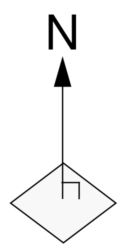


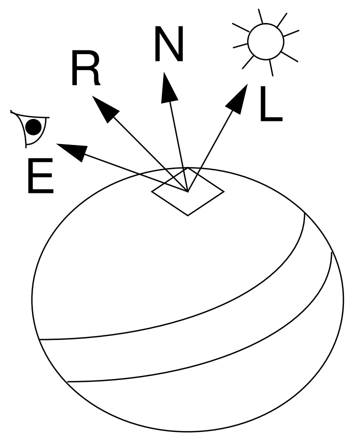

```
glEnable (GL_POINT_SMOOTH), glEnable (GL_LINE_SMOOTH)
```

- in RGBA mode, use alpha values
- in color index mode, use last 4 bits of index

Lighting

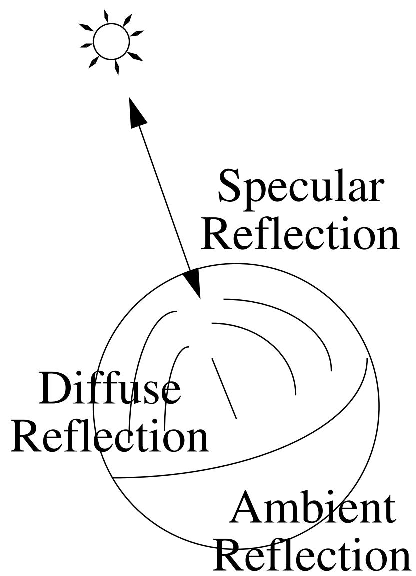
- Approximation of interaction of light and objects
  - where are light source(s) and how do they illuminate the scene?
  - how do objects' material reflect light?
  - how are polygons oriented? (surface normals)
- Phong lighting, not Phong shading
  - colors calculated for each surface normal





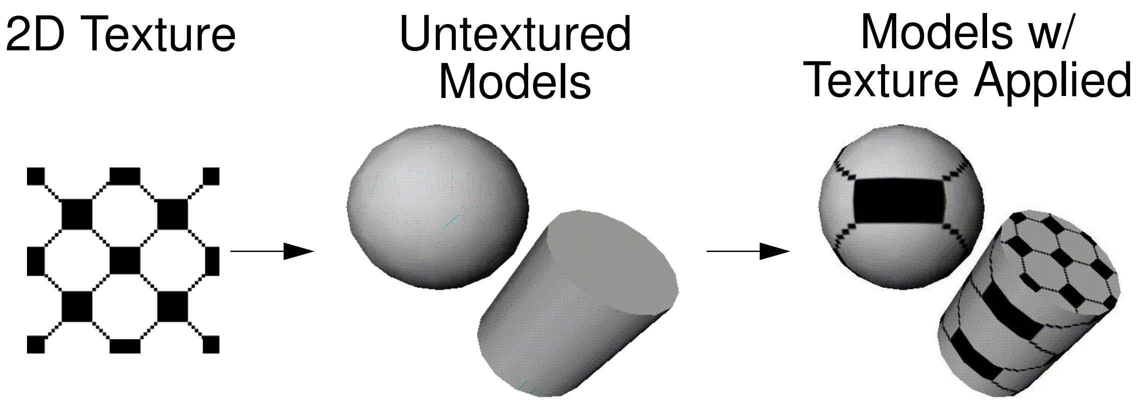
Lighting Properties

- Material Properties of Objects
  - diffuse
  - specular and shininess
  - ambient
  - emission
- Light Source(s)
  - color
  - position
  - local or infinite
  - attenuation (drop off)
  - spot (directional)
- Lighting Model
  - global ambient
  - local or infinite viewer
  - two-sided lighting



Texturing

- Mapping a 2D Image onto a 2D or 3D object
  - Texture Image
  - Texture Coordinates
  - Texture Filter
  - Texture Environment

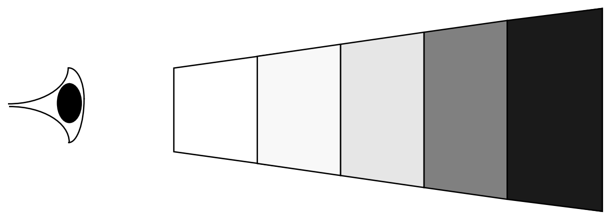


Texture Mapping Uses

- Labels
- Visual Simulation Trick
  - for complex objects
  - draw a “tree” with a simple rectangle
- Reflections
  - environment mapping
- Contouring (depth)
- Antialiased fonts

Atmospheric Effects

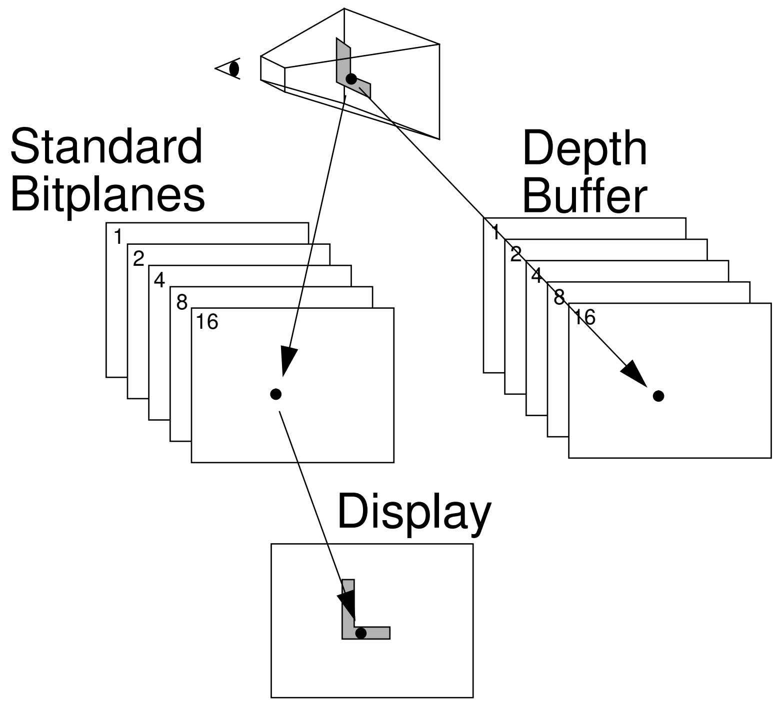
- Density Changes with Distance
- Fog, haze, smoke, and smog are all the same



- Also use for “depth cueing”

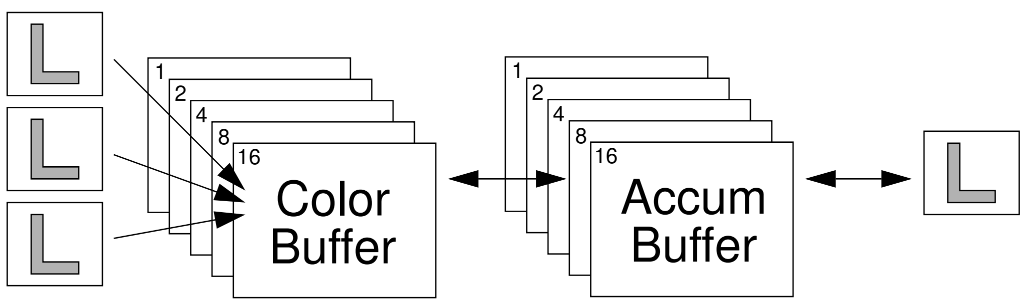
Hidden Surface Removal

- Depth (Z) Buffer
  - depth (Z) value stored for each pixel
  - pixel by pixel comparisons



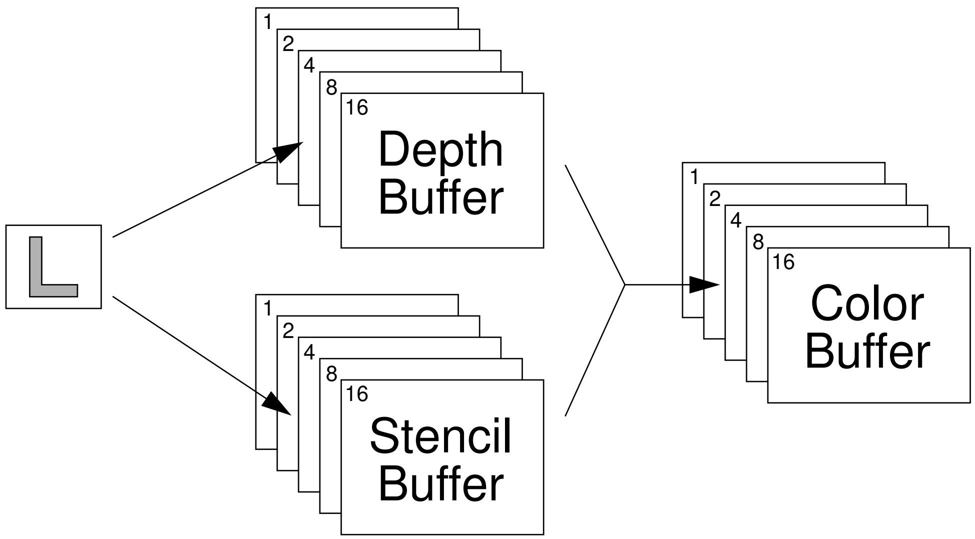
Accumulation Buffer

- Store Multiple Images
  - multiple exposures
- Uses
  - motion blur
  - depth of field (out of focus)
  - scene antialiasing



Stencil Planes

- Additional Pixel Test
- Uses
  - Pixel Masking
  - Capping Solid Geometry



Feedback & Selection

- Usually, transformed vertices and colors generate image in the display buffer
- In feedback mode, the transformed values are returned to the application in an array
- Special tricks for picking objects

Evaluators

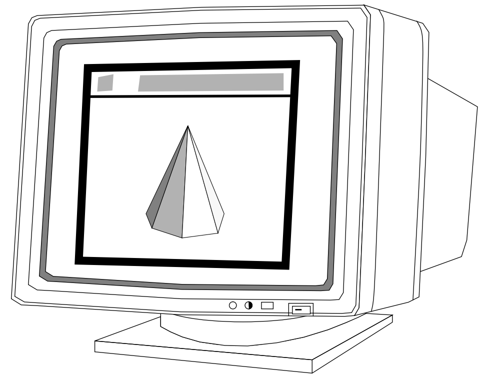
- Support for polynomials (for splines or surfaces)
- Evaluators are foundation for NURBS
- NURBS supported in Utility Library (glu)
  - Non-Uniform Rational B-Splines

Display Lists

- up to now, everything has been in “immediate mode”

Immediate Mode

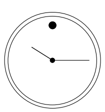
OpenGL Call → Graphics Pipe →



Display List Mode

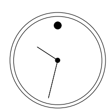
OpenGL Call

↓

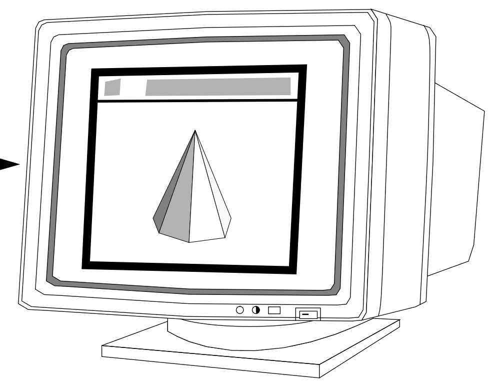


Memory

. . . later that same day →



Graphics Pipe →



Display List Routines

- primary use is for caching routines
  - will see text example
- *may* reside on the server, which can improve performance across network

```
glNewList (GLuint list, GLenum mode)
```

where mode is GL_COMPILE or GL_COMPILE_AND_EXECUTE

```
glEndList ()

glCallList (GLuint list)
```

Display List Editing

- no display list editing, but you can fake it

```
glNewList (1, GL_COMPILE);
    glIndexi (MY_RED);
glEndList ();
glNewList (2, GL_COMPILE);
    glScalef (1.2, 1.2, 1.0);
glEndList ();

glNewList (3, GL_COMPILE);
    glCallList (1);
    glCallList (2);
glEndList ();
    .
    .
glDeleteLists (1, 2);
glNewList (1, GL_COMPILE);
    glIndexi (MY_CYAN);
glEndList ();
glNewList (2, GL_COMPILE);
    glScalef (0.5, 0.5, 1.0);
glEndList ();
```

Text

- minimal direct support
- can access X fonts
- strings may be drawn as a list of display lists
  - the old list base is queried and restored
  - the display list base is shifted to `FONTOFFSET`, so that each ASCII character prints out the appropriate bitmap
- Each character is a display list containing one bitmap

```
#define FONTOFFSET 1000
makeRasterFont(void) {
     GLuint i;
     glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
     for (i = 32; i < 127; i++) {
          glNewList(i+FONTOFFSET, GL_COMPILE);
          glBitmap(8, 13, 0.0, 2.0,
          10.0, 0.0, rasters[i-32]);
          glEndList();
     }
}
```

```
void printString(char *s)
{
     GLuint oldlistbase;
     glGetIntegerv(GL_LIST_BASE, &oldlistbase);
     glListBase(FONTOFFSET);
     glCallLists(strlen(s),
          GL_UNSIGNED_BYTE, (GLubyte *)s);
     glListBase(oldlistbase);
}
```

Summary

- OpenGL API offers sophisticated features
- "State machine" is the OpenGL metaphor
- Geometric primitives, Bitmaps, and Pixel Rectangles are supported
- Each pixel may have many, many bits of information (color, depth, accumulation, and stencil values)
- OpenGL is network interoperable
- OpenGL is window system independent
  - but many hooks to the window system are there

For More Information

- Usenet Group comp.graphics.opengl
- ftp OpenGL Specification and Man Pages from sgigate.sgi.com
  - in pub/opengl directory
  - {gl,glu,glx}.shar.Z files
- Addison-Wesley Publishing

  <u>OpenGL Programming Guide</u>
  - ISBN 0-201-63274-8
  - authors: J. Neider, Davis, and Woo

  <u>OpenGL Reference Manual</u>
  - ISBN 0-201-63276-4
  - author is OpenGL ARB (Architectural Review Board)
- Central marketing & licensing contact: Mason Woo
  - (415) 390-4205; FAX: (415) 964-8671
  - e-mail: woo@sgi.com